Package ‘D3partitionR’

October 12, 2022

**Title**  Interactive Charts of Nested and Hierarchical Data with 'D3.js'

**Version**  0.5.0

**Description**  Builds interactive 'd3.js' hierarchical visualisation easily. D3partitionR makes it easy to build and customize sunburst, circle treemap, treemap, partition chart, ...

**Depends**  R (>= 3.3.1)

**Imports**  data.table, magrittr, htmlwidgets, functional, RColorBrewer, titanic

**License**  AGPL-3

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  6.0.1

**NeedsCompilation**  no


**Maintainer**  Antoine Guillot <antoine.guil@outlook.fr>

**Repository**  CRAN

**Date/Publication**  2017-10-07 14:36:03 UTC

**R topics documented:**

- add_data .................................................. 2
- add_nodes_data ............................................. 3
- add_title .................................................. 3
- aggregate_sessions_to_path .............................. 4
### Description

Append data to a `D3partitionR` object

### Usage

```r
add_data(D3partitionR_object, data, steps, count = "value", color = "name", label = "name", tooltip = "name", aggregate_fun = NULL)
```

### Arguments

- **D3partitionR_object**
  - The `D3partitionR` object to which the data should be appended
- **data**
  - a data.frame object
- **steps**
  - The vector of steps to be used
- **count**
  - The variable to be used as the count variable, typically, the number of occurrences.
- **color**
  - a variable to use as color (default: name)
- **label**
  - a variable to use as label (default: name)
- **tooltip**
  - a variable to use as tooltip (default: name)
- **aggregate_fun**
  - A named list of function which will be used to aggregates to variables used in color, label or tooltips. This only applies to variable in the provided dataset.
add_nodes_data

Value

The D3partitionR object with the appended data

Description

Add informations (for instance new names, colors, ....) to the nodes of a D3_partitionR object

Usage

add_nodes_data(D3partitionR_object, nodes_data)

Arguments

D3partitionR_object
The D3partitionR object to which the data should be appended

nodes_data
a names list where the name of each element is the name of a node. The data will be appended to the node in the nested list

Value

The D3partitionR object with the appended nodes data

add_title

Add a title to a D3partitionR object

Description

Add a title to a D3partitionR object

Usage

add_title(D3partitionR_object, text, style = NULL)

Arguments

D3partitionR_object
The D3partitionR object to which the data should be appended

text
Title text

style
A valid CSS string which will be applied to the title

Value

A D3partitionR object
aggregate_sessions_to_path

*Aggregate a data.frame in long format with a column containing steps of each session. For instance the function can be used with a frame of the form Unique ID - Step - Value 1 - ... - Value N*

**Description**

Aggregate a data.frame in long format with a column containing steps of each session. For instance the function can be used with a frame of the form Unique ID - Step - Value 1 - ... - Value N.

**Usage**

```r
aggregate_sessions_to_path(data, step_col = "step", id_col = "ID", values_cols = NULL, agg_function_path = sum, agg_function_session = sum, sep = "->")
```

**Arguments**

- `data`: A dataframe.
- `step_col`: The name of the column containing the steps. The steps are assumed to be ordered.
- `id_col`: Column containing the unique identifier of each session.
- `values_cols`: Names of the other columns to keep. Default: NULL.
- `agg_function_path`: Aggregation function on a path level.
- `agg_function_session`: Aggregation function on a session level.
- `sep`: String used to separate the different steps. Default: "->".

**Value**

A data.table with the columns specified in `count_col`, `values_cols` and one column per step in the path.

---

**compile_D3_partitionR**

*Compile D3partitionR object to plot it*

**Description**

Compile D3partitionR object to plot it.

**Usage**

```r
compile_D3_partitionR(D3partitionR_object)
```
**compute_unique_leaf_name**

**Arguments**

D3partitionR_object
The D3partitionR object to which the data should be appended

**Value**

A D3partitionR compiled object

---

**compute_unique_leaf_name**

*Return all the leaf names*

---

**Description**

Return all the leaf names

**Usage**

compute_unique_leaf_name(nested_list)

**Arguments**

nested_list A nested_list where each node has a name attribute

---

**D3partitionR**

*Creates a D3partitionR object*

---

**Description**

Creates a D3partitionR object

**Usage**

D3partitionR()

**Value**

A blank D3partitionR object (S3 class)
D3partitionR-shiny  

_Shiny bindings for D3partitionR_

**Description**

Output and render functions for using D3partitionR within Shiny applications and interactive Rmd documents.

**Usage**

```r
D3partitionROutput(outputId, width = "100%", height = "400px")

renderD3partitionR(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

- `outputId`: output variable to read from
- `width`, `height`: Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
- `expr`: An expression that generates a D3partitionR
- `env`: The environment in which to evaluate `expr`.
- `quoted`: Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

---

df_to_nest  

_Transform a dataframe to a nested lists structure (i.e. hierarchical)._

**Description**

Transform a dataframe to a nested lists structure (i.e. hierarchical).

**Usage**

```r
df_to_nest(data, step_cols, nodes_data = NULL, count_col = "value", value_cols = NULL, agg_function = sum, na_behavior = "rm")
```

**Arguments**

- `data`: The data frame to convert to the nested structure. It needs to have several columns, each ones account for a given step
- `step_cols`: vector containing the names of the columns which should be used as steps. The vector should be ordered. ex: c('step1','step2','step3')
- `nodes_data`: A named list to add addition informations to each nodes
**find_min_max_tree**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count_col</td>
<td>Number of occurrences in this path (succession of steps). Default: NULL.</td>
</tr>
<tr>
<td>value_cols</td>
<td>Names of the other columns to keep. Default: NULL.</td>
</tr>
<tr>
<td>agg_function</td>
<td>Aggregation function to be applied to value_cols. Ex: mean, sum. Default: sum. Weighted version can also be used, the weighting will be done using the counting variable</td>
</tr>
<tr>
<td>na_behavior</td>
<td>How to deal with missing data?</td>
</tr>
</tbody>
</table>

**Value**

A data.table with the columns specified in count_col, value_cols and one column per step in the path.

---

**find_min_max_tree**

*Find the maximum values of a given var in a tree*

**Description**

Find the maximum values of a given var in a tree.

**Usage**

```
find_min_max_tree(nested_list, variable = "value")
```

**Arguments**

- `nested_list`: A nested_list where each node has a name attribute.
- `variable`: A nested_list where each node has a name attribute.

---

**get_all_nodes_names**

*Return all the possible nodes names*

**Description**

Return all the possible nodes names.

**Usage**

```
get_all_nodes_names(nested_list, variable = "name")
```

**Arguments**

- `nested_list`: A nested_list where each node has a name attribute.
- `variable`: The variable to collect.
is_present_variable  
*Check if a variable is present in a D3partitionR object*

**Description**

Check if a variable is present in a D3partitionR object

**Usage**

```r
is_present_variable(variable, D3partitionR_object)
```

**Arguments**

- `variable`  
The variable which presence is to be checked
- `D3partitionR_object`  
The D3partitionR object

**Value**

TRUE/FALSE

---

plot.D3partitionR  
*Plot D3partitionR object*

**Description**

Plot D3partitionR object

**Usage**

```r
# S3 method for class 'D3partitionR'
plot(x, width = NULL, height = NULL,
     elementId = NULL, sizingPolicy = NULL, ...)
```

**Arguments**

- `x`  
A D3partitionR object to plot
- `width`  
width of the widget in pixel/percent
- `height`  
height of the widget in pixel/percent
- `elementId`  
html id of the widget
- `sizingPolicy`  
sizing policy
- `...`  
parameters for method consistency
Examples

```r
require(titanic)
require(data.table)
## Reading data
titanic_data = data.table(titanic::titanic_train)
## Aggregating data to have unique sequence for the 4 variables
var_names=c('Sex','Embarked','Pclass','Survived')
data_plot=titanic_data[,N,by=var_names]
data_plot[,,(var_names):=lapply(var_names,function(x){data_plot[[x]]=paste0(x,' ',data_plot[[x]])})]
## Plotting the chart
library("magrittr")
d3=D3partitionR() %>%
  add_data(data_plot,count = 'N',steps=c('Sex','Embarked','Pclass','Survived')) %>%
  add_title('Titanic')
## Not run:
plot(d3)
## End(Not run)
```

---

**scale_type**

*Check if the scale variable is discrete or continuous*

**Description**

Check if the scale variable is discrete or continuous

**Usage**

```r
scale_type(color_variable, D3partitionR_object)
```

**Arguments**

- `color_variable` The color variable to be assessed
- `D3partitionR_object` The D3partitionR object

**Value**

TRUE/FALSE
set_continuous_color_scale

Add a custom discrete color scale

Description
Add a custom discrete color scale

Usage
set_continuous_color_scale(D3partitionR_object, color_palette)

Arguments
D3partitionR_object
The D3partitionR object to which the data should be appended

color_palette
a vector of two colors, the first one is use on the bottom of the scale, the other on the top.

Value
A D3partitionR object

set_chart_type
Set the chart_type

Description
Set the chart_type

Usage
set_chart_type(D3partitionR_object, chart_type)

Arguments
D3partitionR_object
The D3partitionR object to which the data should be appended

chart_type
type fo chart to use (in c('sunburst','treemap','circle_treemap','partition_chart','icicle'))

Value
A D3partitionR object
set_discrete_color_scale

Add a custom discrete color scale

Description
Add a custom discrete color scale

Usage
set_discrete_color_scale(D3partitionR_object, color_palette)

Arguments
- D3partitionR_object
  The D3partitionR object to which the data should be appended
- color_palette
  A vector (or a named vector with levels of the variable color)

Value
A D3partitionR object

set_labels_parameters
Set the labels parameters

Description
Set the labels parameters

Usage
set_labels_parameters(D3partitionR_object, visible = T, cut_off = 3, style = NULL)

Arguments
- D3partitionR_object
  The D3partitionR object to which the data should be appended
- visible
  boolean, should the labels be displayed? Default: TRUE
- cut_off
  a numeric variable between 0 and 100. Nodes which represent less than cut_off percents of the current root will have their labels hidden.
- style
  a valid CSS string to be applied to the labels. Default: NULL

Value
A D3partitionR object
**set_legend_parameters**  
*Set the legend parameter*

**Description**
Set the legend parameter

**Usage**
```r
set_legend_parameters(D3partitionR_object, visible = T, zoom_subset = F, width = 100)
```

**Arguments**
- **D3partitionR_object**: The D3partitionR object to which the data should be appended
- **visible**: boolean, should the trail be displayed? Default: TRUE
- **zoom_subset**: boolean, if TRUE, only the modalities present in the children of the zoomed root are displayed in the legend.
- **width**: legend width in pixel

**Value**
A D3partitionR object

---

**set_shiny_input**  
*Configuration of a D3partitionR object as a Shiny input*

**Description**
Configuration of a D3partitionR object as a Shiny input

**Usage**
```r
set_shiny_input(D3partitionR_object, input_id, enabled_inputs = list(clicked_node = T, leaves = T, nodes = T, ancestors = T, children_path = F))
```

**Arguments**
- **D3partitionR_object**: The D3partitionR object to which the data should be appended
- **input_id**: The id of the input
- **enabled_inputs**: which inputs should be enabled? default to list(clicked_node=T,leaves=T,nodes=T,ancestors=T,child_path=F)
set_tooltip_parameters

Set the tooltips parameter

Description
Set the tooltips parameter

Usage

set_tooltip_parameters(D3partitionR_object, visible = T, style = NULL, 
builder = "table")

Arguments

D3partitionR_object
The D3partitionR object to which the data should be appended
visible
boolean, should the trail be displayed? Default: TRUE
style
a valid CSS string to be applied to the tooltip. Default: NULL
builder
Tooltip builder to use for the tooltip. Can either one of the predefined tooltip
('table','basic') or a js expression returning a tooltip.

Value
A D3partitionR object

set_trail
Enable/disable the trail of steps

Description
Enable/disable the trail of steps

Usage

set_trail(D3partitionR_object, visible = T)

Arguments

D3partitionR_object
The D3partitionR object to which the data should be appended
visible
boolean, should the trail be displayed? Default: TRUE
strip_path

Strip a dataframe containing a step into separate columns

Description
Strip a dataframe containing a step into separate columns

Usage

```r
strip_path(data, path_col = "path", count_col = "count",
value_cols = NULL, sep = "->")
```

Arguments

- **data**: A dataframe containing the path.
- **path_col**: Name of the column containing the path. The path should be a string of the format "step 1 -> step 2 -> step 3". Default: "path"
- **count_col**: Name of the column containing the number of occurrences of the path. Default: "count"
- **value_cols**: Names of the other columns to keep. Default: NULL
- **sep**: String used to separate the different steps. Default: "->"

Value

A data.table with the columns specified in count_col, value_cols and one column per step in the path

tooltip_builder

Build tooltip html function

Description
Build tooltip html function

Usage

```r
tooltip_builder(type)
```

Arguments

- **type**: a tooltip type: 'basic' (i.e the variable value) or 'table'(i.e. a table with the variables names and value)
Index

add_data, 2
add_nodes_data, 3
add_title, 3
aggregate_sessions_to_path, 4

compile_D3_partitionR, 4
compute_unique_leaf_name, 5

D3partitionR, 5
D3partitionR-shiny, 6
D3partitionROutput
  (D3partitionR-shiny), 6
df_to_nest, 6

find_min_max_tree, 7
get_all_nodes_names, 7
is_present_variable, 8

plot.D3partitionR, 8

renderD3partitionR
  (D3partitionR-shiny), 6

scale_type, 9
set_chart_type, 10
set_continuous_color_scale, 10
set_discrete_color_scale, 11
set_labels_parameters, 11
set_legend_parameters, 12
set_shiny_input, 12
set_tooltip_parameters, 13
set_trail, 13
strip_path, 14

tooltip_builder, 14